III. TUTORIAL BÁSICO DE SYMBOLIC

El *toolbox symbolic* de MATLAB, contiene un amplio conjunto de funciones muy útiles para realizar cálculos simbólicos. Un listado completo de las mismas puede obtenerse ejecutando sobre la línea de comandos del *workspace* la sentencia:

```
help toolbox\symbolic
```

Este apunte presenta, sobre un ejemplo de aplicación, la utilización de las funciones básicas del *toolbox symbolic*. Para una descripción más detallada de las mismas remitimos al *Help* o bien a la *Guía del Usuario* correspondiente.

Cabe aclarar que las funciones que aquí se ejemplifican corresponden a la versión 4.2 de MATLAB. Versiones posteriores han modificado la sintaxis de algunas de ellas.

Consideremos un sistema descripto por el siguiente modelo dinámico no lineal:

$$\dot{x}_1 = x_2 + x_1 + u$$
$$\dot{x}_2 = \sqrt{1 + x_1} - \sqrt{1 + x_2}$$
$$y = \sqrt{x_1} + \sqrt{x_2}$$

y supongamos que queremos obtener el punto estático de operación de este sistema como función del control *u*, para posteriormente linealizarlo en torno a un valor particular del mismo. Igualando a cero las expresiones de las

derivadas de los estados, nos queda el siguiente sistema de ecuaciones algebraico:

$$0 = x_2 + x_1 + u$$
$$0 = \sqrt{1 + x_1} - \sqrt{1 + x_2}$$

que resolvemos utilizando la función *solve*:

```
[x1o,x2o]=solve('x2+x1+u','sqrt(x1+1)-sqrt(x2+1)',
'x1,x2')
```

donde los dos primeros argumentos de la función corresponden a las expresiones de las derivadas de los estados y el último a las variables que se pretende despejar. En el caso en que las ecuaciones están descriptas por un solo miembro, el programa asume que el otro miembro es idéntico a cero. La ejecución del comando anterior retorna dos puntos de funcionamiento estático coincidentes, dados por:

```
x1o =
[-1/2*u]
[-1/2*u]

x2o =
[-1/2*u]
[-1/2*u]
```

Es conocido que la linealización alrededor de un punto de operación (x_{1o}, x_{2o}) de un sistema no lineal, tiene las matrices A, B y C definidas como:

$$A = \left[\begin{array}{cc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{array} \right]_{\substack{x_1 = x_{1o} \\ x_2 = x_{2o}}}$$

$$B = \left[\begin{array}{c} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{array} \right]_{\substack{x_1 = x_{1o} \\ x_2 = x_{2o}}}$$

$$C = \left[\begin{array}{cc} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} \end{array} \right]_{\substack{x_1 = x_{1o} \\ x_2 = x_{2o}}}$$

Para calcular estas matrices podemos apelar a tres funciones diferentes que son: *diff*, *taylor* y *jacobian*. A continuación se utiliza la función *jacobian* para obtener las matrices A y C, y la función *diff* para obtener la matriz B (esta última no puede calcularse a través de *jacobian* por incompatibilidad de dimensiones):

```
A=jacobian(sym('x2+x1+u;sqrt(x1+1)-sqrt(x2+1)'),
sym('x1,x2'))
B=diff(sym('x2+x1+u;sqrt(x1+1)-sqrt(x2+1)'),
sym('u'))
C=jacobian(sym('sqrt(x1)+sqrt(x2)'),sym('x1,x2'))
```

Ejecutando estos comandos el programa retorna:

```
A =
[                1,                   1]
[1/2/(1+x1)^(1/2), -1/2/(x2+1)^(1/2)]

B =
[1]
[0]

C =
[1/2/x1^(1/2), 1/2/x2^(1/2)]
```

NOTA: las funciones anteriores también pueden ejecutarse definiendo previamente el vector que se desea derivar utilizando el comando 'sym'. Esto es:

```
F=sym('[x2+x1+u;sqrt(x1+1)-sqrt(x2+1)']) 
A=jacobian(F,sym('x1,x2'))
B=diff(F,sym('u'))
```

Finalmente para determinar las matrices A, B y C correspondientes a la linealización del sistema en torno al punto de operación, debemos reemplazar en ellas la aparición de las variables x_1 y x_2 por los valores almacenados en x_{1o} y x_{2o} obtenidos anteriormente. Para ello se emplea el comando *subs*:

```
A=subs(A,'-1/2*u','x1')
A=subs(A,'-1/2*u','x2')
```

```
C=subs(C,'-1/2*u','x1')
C=subs(C,'-1/2*u','x2')
```

NOTA: cabe aclarar que de no haber obtenido soluciones múltiples para x10 y x20, podríamos escribir las sentencias anteriores como:
A=subs(A,x1o,'x1')

Ejecutando, obtenemos:

```
A =
[                  1,                       1]
[1/2/(1-1/2*u)^(1/2), -1/2/(1-1/2*u)^(1/2)]

C =
[1/2/(-1/2*u)^(1/2), 1/2/(-1/2*u)^(1/2)]
```

Para observar la expresión de cualquiera de estas matrices A, B y C en una forma mas elaborada, podemos utilizar el comando *pretty*:

```
pretty(A)
```

retornando:

```
  ÇÊ        1                    1         ÍÐ
  ÈË                                       ÎÑ
  ÈË        1                    1         ÎÑ
  ÈË õõõõõõõõõõõõõõõõõõ  - õõõõõõõõõõõõõõõõõõ ÎÑ
  ÈË              1/2                  1/2  ÎÑ
  ÉÌ 2(1 - 1/2 u)        2 (1 - 1/2 u)    ÏÒ
```

De esta manera hemos obtenido las expresiones simbólicas de las matrices A, B y C correspondientes a la linealización de un modelo, en torno a un punto de operación.

A fin de ejemplificar la utilización de algunas otras funciones importantes, consideremos la diagonalización del modelo lineal obtenido. Como es conocido, para diagonalizar debemos aplicar la transformación lineal dada por la matriz de los autovectores de A, la cual puede hallarse a través de la función *eigensys*:

```
[T,P] = eigensys(A)
```

Esta función retorna una matriz de autovectores [T] y un vector con los autovalores del sistema [P].

NOTA: si la matriz A aún no se encontrara definida, esta función puede ejecutarse, como:
*[T,P] = eigensys(sym('[1, 1;1/2/(1-1/2*u)^(1/2), -1/2/(1-1/2*u)^(1/2)]'))*

Para encontrar las matrices A_d, B_d y C_d del modelo lineal diagonalizado, debemos realizar la siguiente operatoria:

$$A_d = T^{-1} \cdot A \cdot T$$
$$B_d = T^{-1} \cdot B$$
$$C_d = C \cdot T$$

Inicialmente tenemos que hallar la matriz inversa de T, para lo cual usamos el comando *inverse*. Esto es:

```
TI=inverse(T)
```

Luego, para obtener A_d la multiplicación de matrices se realiza con la función *symmul*. Como esta función toma únicamente dos argumentos, para obtener la expresión por medio de una sola línea de comando, se realiza una anidación de funciones *symmul*:

```
Ad=symmul(TI,symmul(A,T))
```

De igual manera obtenemos Bd y Cd como:

```
Bd=symmul(TI,B)
Cd=symmul(C,T)
```

*NOTA: otras operaciones simbólicas básicas se realizan por medio de las funciones 'symadd', 'symsub', 'symdyv', 'sympow', etc. Versiones de Matlab posteriores a la 4.2 eliminaron estas funciones reemplazándolas por los operandos matemáticos comunes (+, -, *, ^).*

Utilizando otras operaciones matriciales como las mencionadas en la nota anterior, es inmediato obtener la matriz de transferencia del sistema lineal (este cálculo puede hacerse con el sistema diagonalizado o no).

$$T(s) = C \cdot (s \cdot I - A)^{-1} \cdot B$$

Para ello debemos crear inicialmente una matriz identidad simbólica de dimensiones adecuadas, y luego anidar las demás operaciones en una única sentencia. Esto es:

```
I=sym('[1,0;0,1]')
```

```
Ts=symmul(symmul(C,inverse(symsub(symmul(sym('s'
),I),A))),B)
```

Ejecutando esta sentencia se obtiene una expresión larga y complicada de la función de transferencia. Para que sea mostrada en un formato más compacto podemos acudir a la función de simplificación *simple* y a la conocida *pretty*. Esto es, haciendo:

```
Ts=simple(Ts)
pretty(Ts)
```

obtenemos:

```
                     1/2              1/2
                    2    (s (4 - 2 u)    + 2)
1/2 õõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõõ
            1/2    2         1/2                  1/2
       (- u)    (s  (4 - 2 u)    + s - s (4 - 2 u)    - 2)
```

También podríamos obtener el polinomio característico del modelo linealizado a través de la función *charpoly*. Esto es:

```
PolCar=Charpoly(A)
```

Retornando:

```
PolCar =
(x^2*(4-2*u)^(1/2)+x-x*(4-2*u)^(1/2)-2)/(4-
2*u)^(1/2)
```

Finalmente si quisiéramos obtener las matrices A y C en forma numérica para un determinado valor de *u*, por ejemplo *u=-3*, empleamos las funciones *subs* para substituir la expresión de u por el valor deseado y *numeric* para realizar las operaciones algebraicas. Esto es:

```
An=subs(A,-3,'u')
Bn=subs(B,-3,'u')
Cn=subs(C,-3,'u')
```

Retornando:

```
An =
[                    1,                    1]
[1/10*5^(1/2)*2^(1/2), -1/10*5^(1/2)*2^(1/2)]

Bn =
[1]
[0]

Cn =
[1/6*3^(1/2)*2^(1/2), 1/6*3^(1/2)*2^(1/2)]
```

Luego aplicamos la función *numeric*:

```
An=numeric(An)
Bn=numeric(Bn)
Cn=numeric(Cn)
```

Retornando:

An =

1.0000 1.0000

0.3162 -0.3162

Bn =

1

0

Cn =

0.4082 0.4082

NOTA: el toolbox symbolic posee un conjunto de funciones de simplificación ('simple', 'simplify', 'factor', 'expand', 'collect') que resultan muy útiles a la hora de reducir las expresiones que se obtienen al aplicar otras operaciones simbólicas. Es recomendable su uso frecuente en pasos intermedios del cálculo.